(12) UK Patent Application (19) GB (11) 2 396 442 (13) A

(43) Date of A Publication 23.06.2004

(21) Application No: 0228438.8

(22) Date of Filing: 05.12.2002

(30) Priority Data:
(31) 0221562 (32) 17.09.2002 (33) GB

(71) Applicant(s):
Micron Technology, Inc.
(Incorporated in USA - Delaware)
8000 South Federal Way, Boise,
Idaho 83706-9632,
United States of America

(72) Inventor(s):
Graham Kirsch

(74) Agent and/or Address for Service:
Carpmaels & Ransford
43 Bloomsbury Square, LONDON,
WC1A 2RA, United Kingdom

(51) INT CL[7]:
G06F 15/78 , G11C 7/10

(52) UK CL (Edition W):
G4A AMP AMX

(56) Documents Cited:
EP 0584783 A2 **WO 2001/001242 A1**
US 5956274 A **US 5953738 A**
IEEE Transactions on VLSI Systems, Volume 4, Number 1, pages 56-69, 1996

(58) Field of Search:
UK CL (Edition W) **G4A**
INT CL[7] **G06F**
Other: **Online: WPI, EPODOC, PAJ, INSPEC, XPESP, Internet**

(54) Abstract Title: **Host memory interface for a parallel processor**

(57) A memory interface 112 for a parallel processor 110 which has an array of processing elements and can receive a memory address and supply the memory address to a memory 106 connected to the processing elements. The processing elements transfer data to and from the memory at the memory address. The memory interface can connect to a host 102 configured to access data in a conventional SDRAM memory device so that the host can access data in the memory. The invention also relates to an active memory which includes memory 106, an array of processing elements 110, and the memory interface 112. The memory interface enables a host which is configured to connect to conventional "non-active" memory devices to access an active memory, and also permits different types of host processors to access the memory in the active memory device.

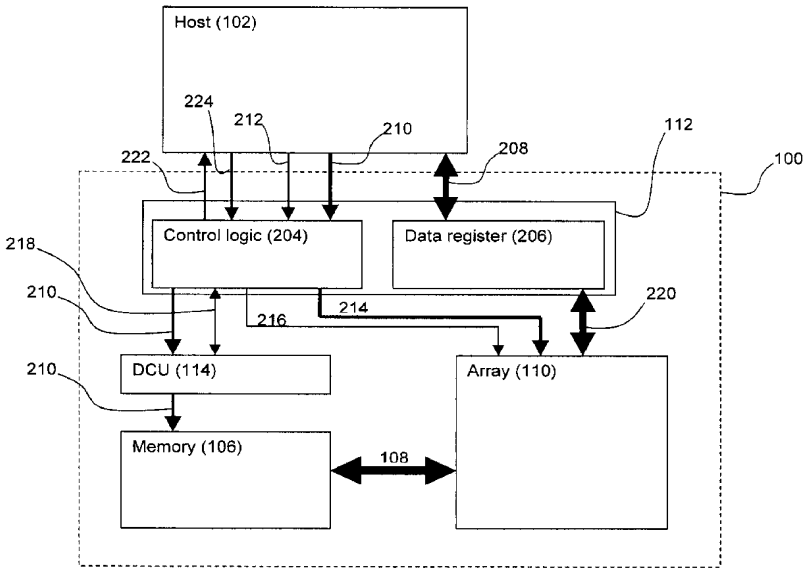
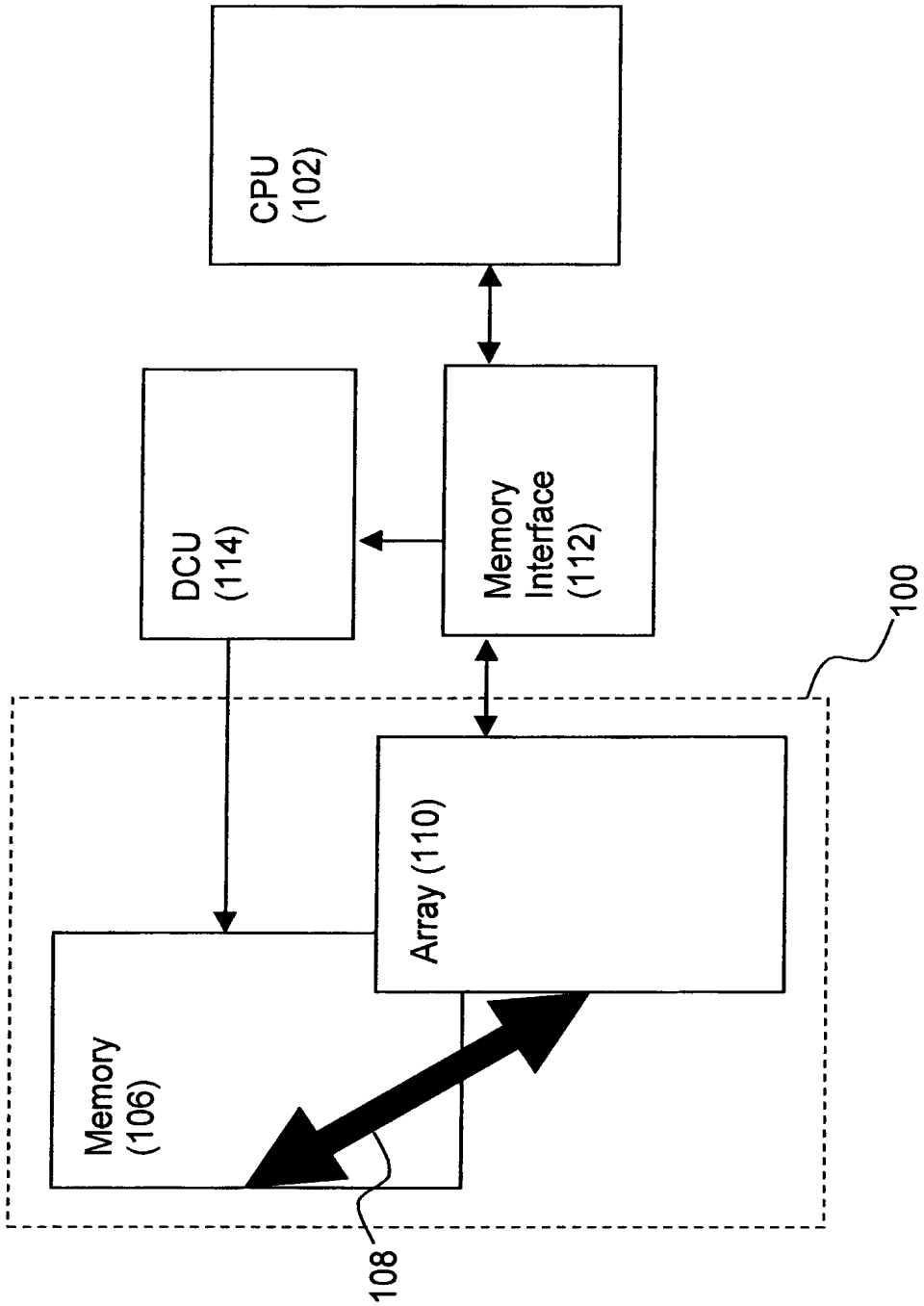


FIG. 2

GB 2 396 442 A

FIG. 1


CPU
(102)
DCU
(114)
Memory
Interface
(112)
100
Array (110)
Memory
(106)
108

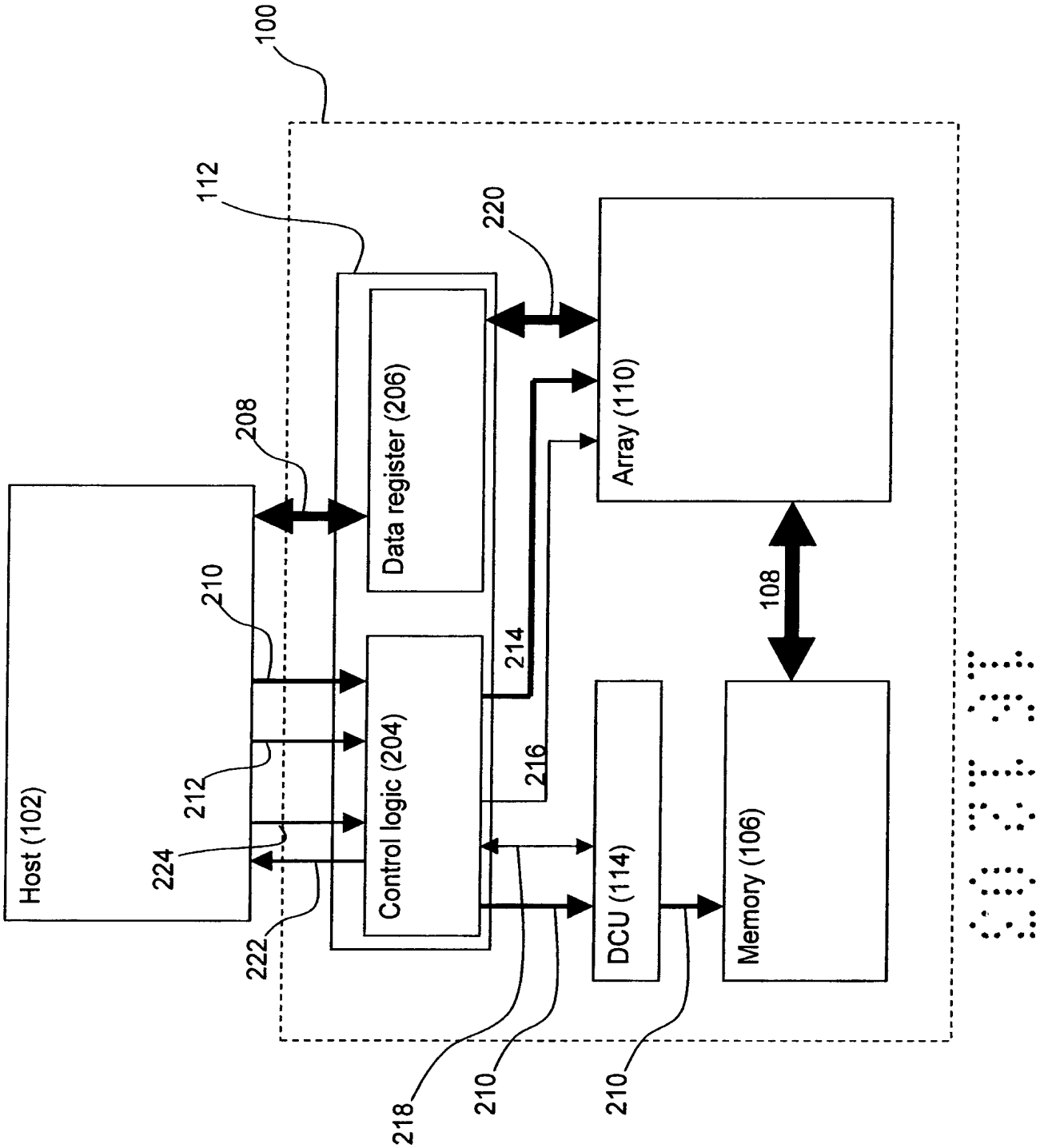

100
112
220
208
Data register (206)
Array (110)
210
214
108
Host (102)
212
Control logic (204)
216
224
DCU (114)
Memory (106)
222
218
210
210

FIG. 2

FIG. 3

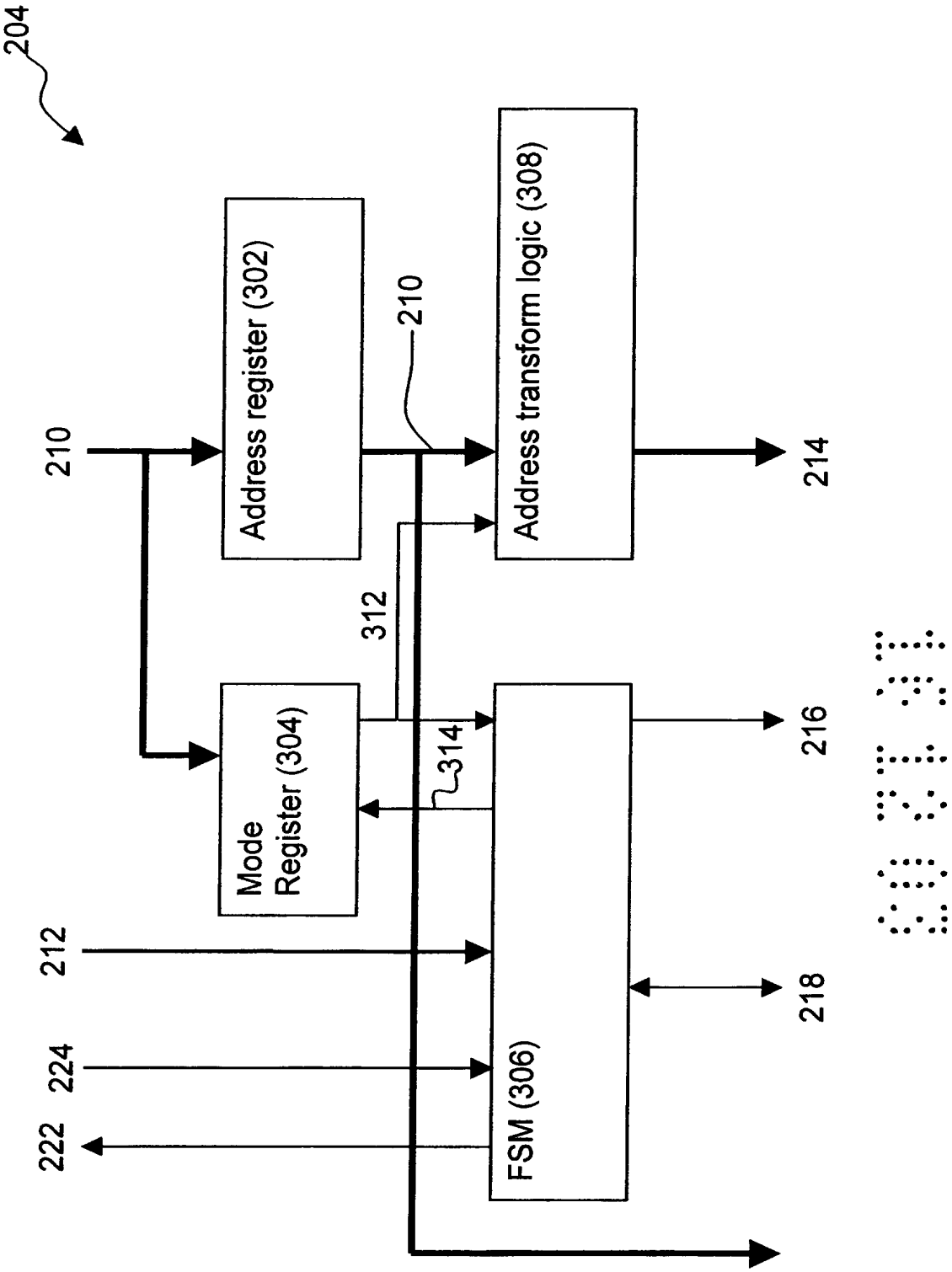
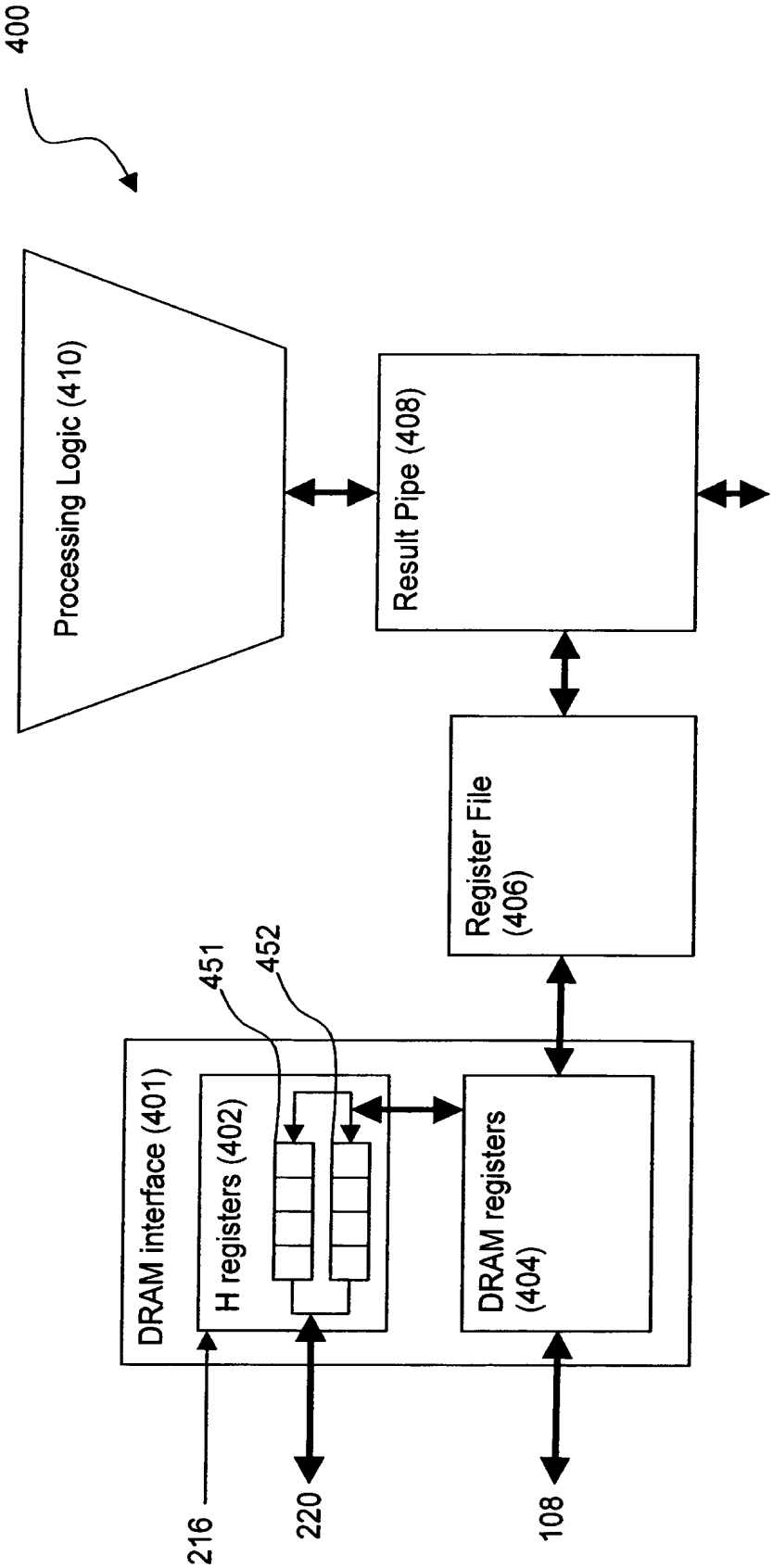

204
210
Address register (302)
210
Address transform logic (308)
214
312
Mode Register (304)
314
216
212
FSM (306)
218
224
222

FIG. 4


400
Processing Logic (410)
Result Pipe (408)
Register File (406)
DRAM interface (401)
H registers (402)
DRAM registers (404)
451
452
216
220
108

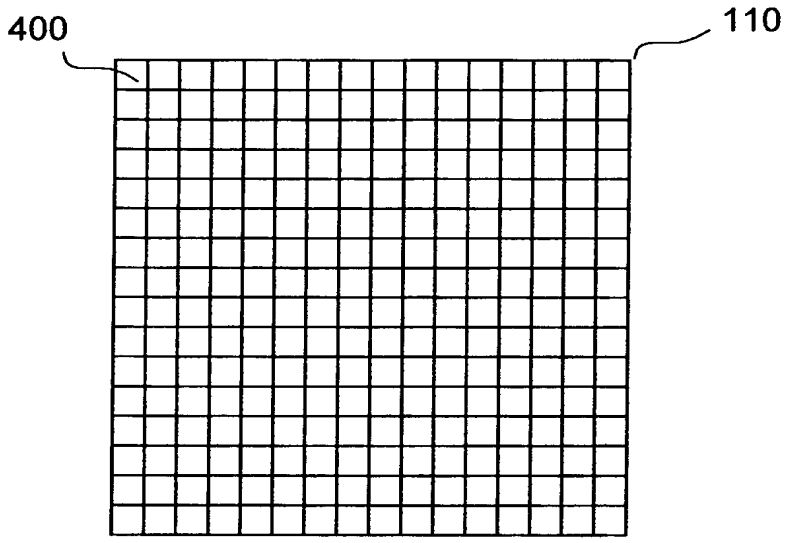


FIG. 5a

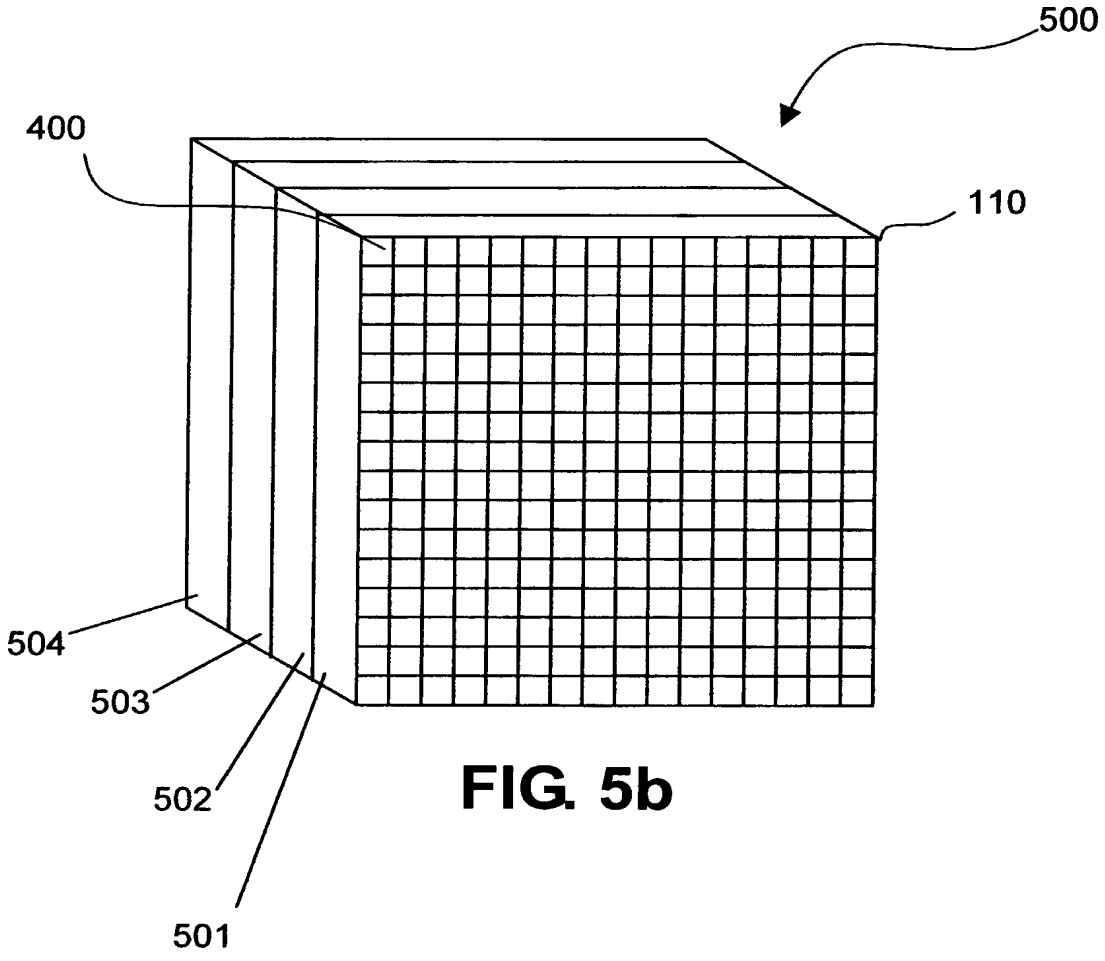


FIG. 5b

FIG. 6a

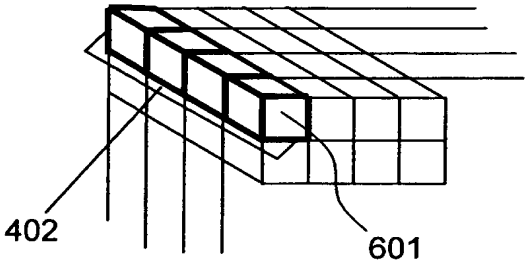


FIG. 6b

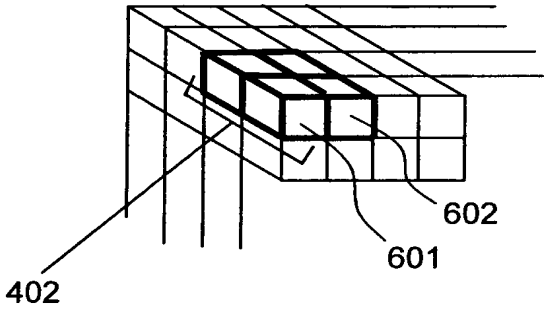


FIG. 6c

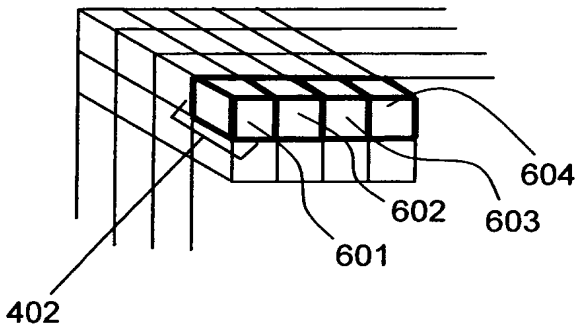
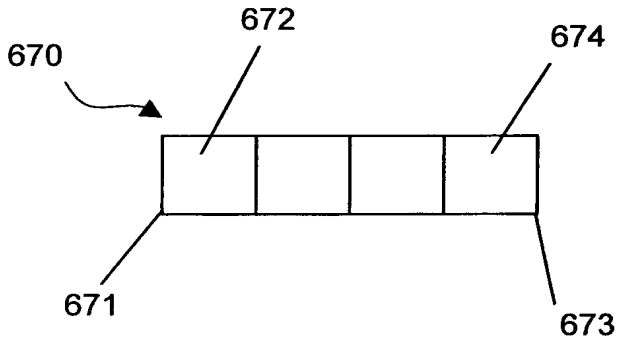


FIG. 6d



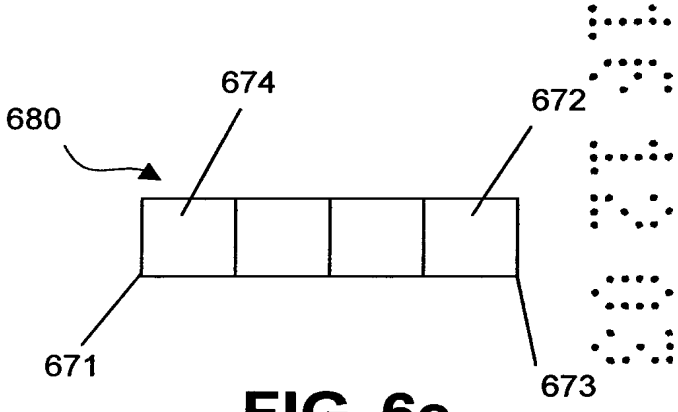


FIG. 6e

FIG. 7

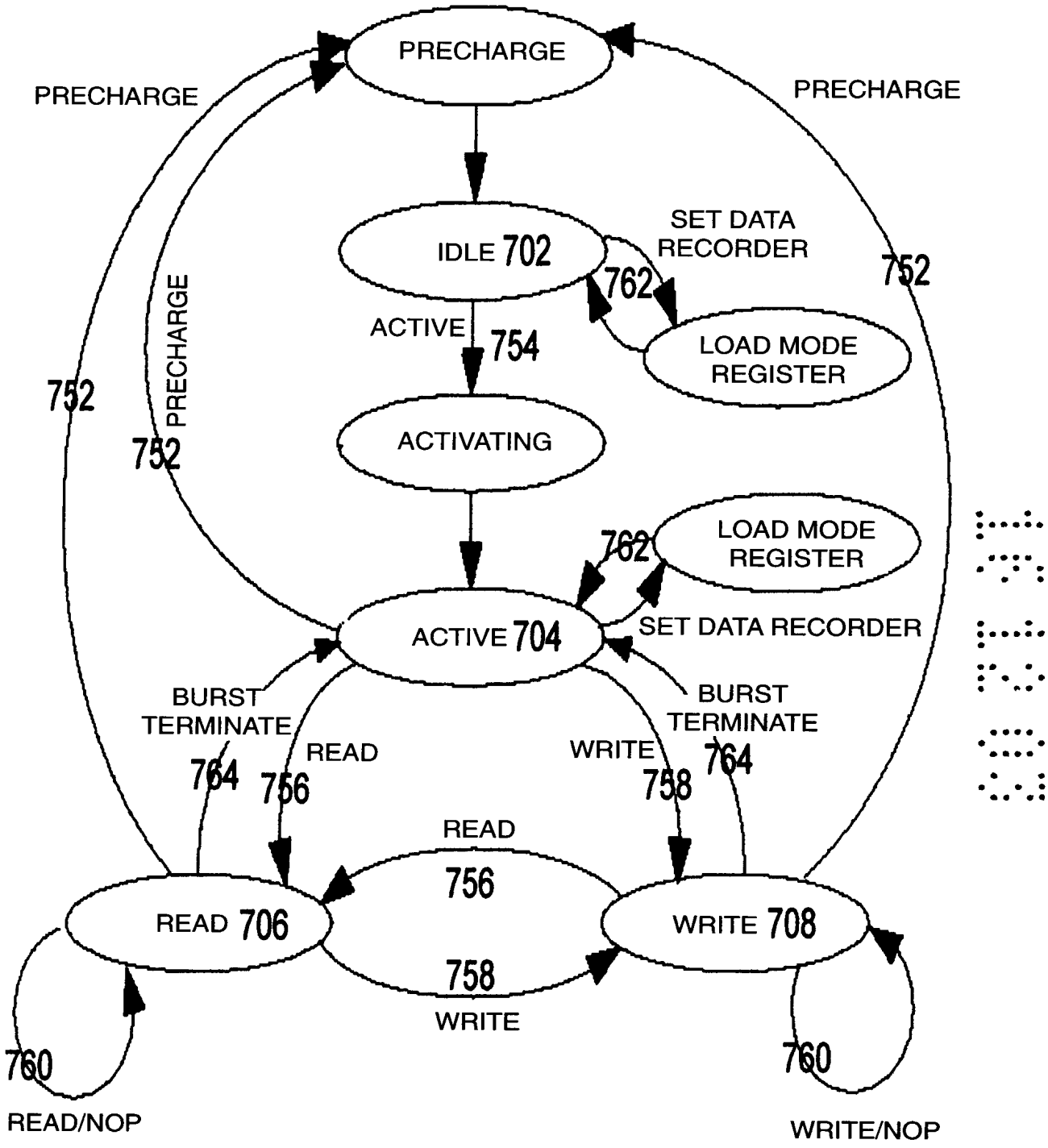

PRECHARGE
PRECHARGE
PRECHARGE
IDLE 702
SET DATA RECORDER
762
752
ACTIVE
754
LOAD MODE REGISTER
752
PRECHARGE
752
ACTIVATING
LOAD MODE REGISTER
762
ACTIVE 704
SET DATA RECORDER
BURST TERMINATE
764
READ
756
WRITE
758
BURST TERMINATE
764
READ
756
READ 706
WRITE 708
758
WRITE
760
760
READ/NOP
WRITE/NOP

FIG. 8

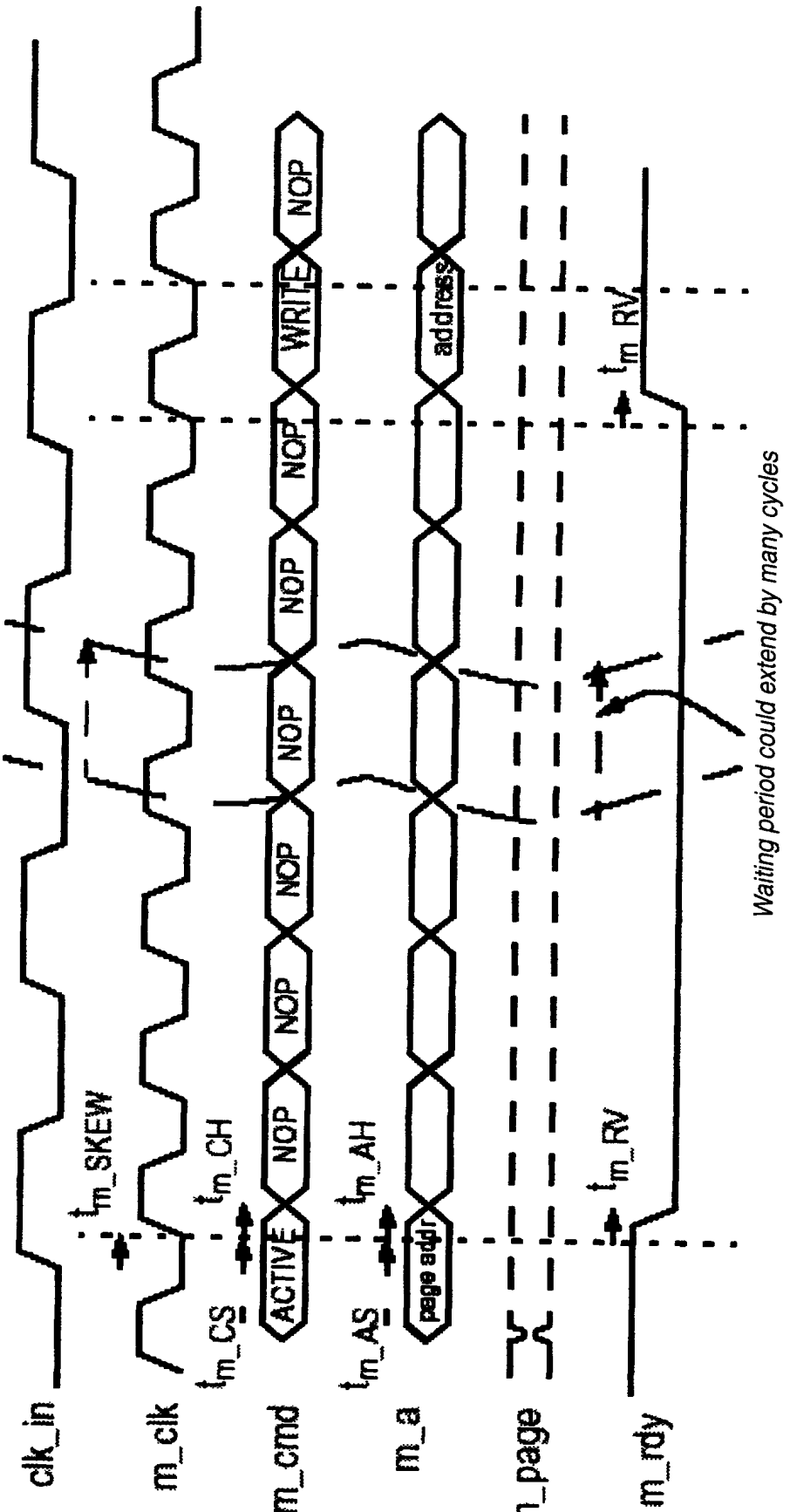

clk_in
m_clk
m_cmd
m_a
m_page
m_rdy
t_{m_SKEW}
t_{m_CH}
t_{m_CS}
t_{m_AH}
t_{m_AS}
t_{m_RV}
t_{m_RV}
ACTIVE
NOP
NOP
NOP
NOP
NOP
WRITE
NOP
page adr
address
Waiting period could extend by many cycles

FIG. 9

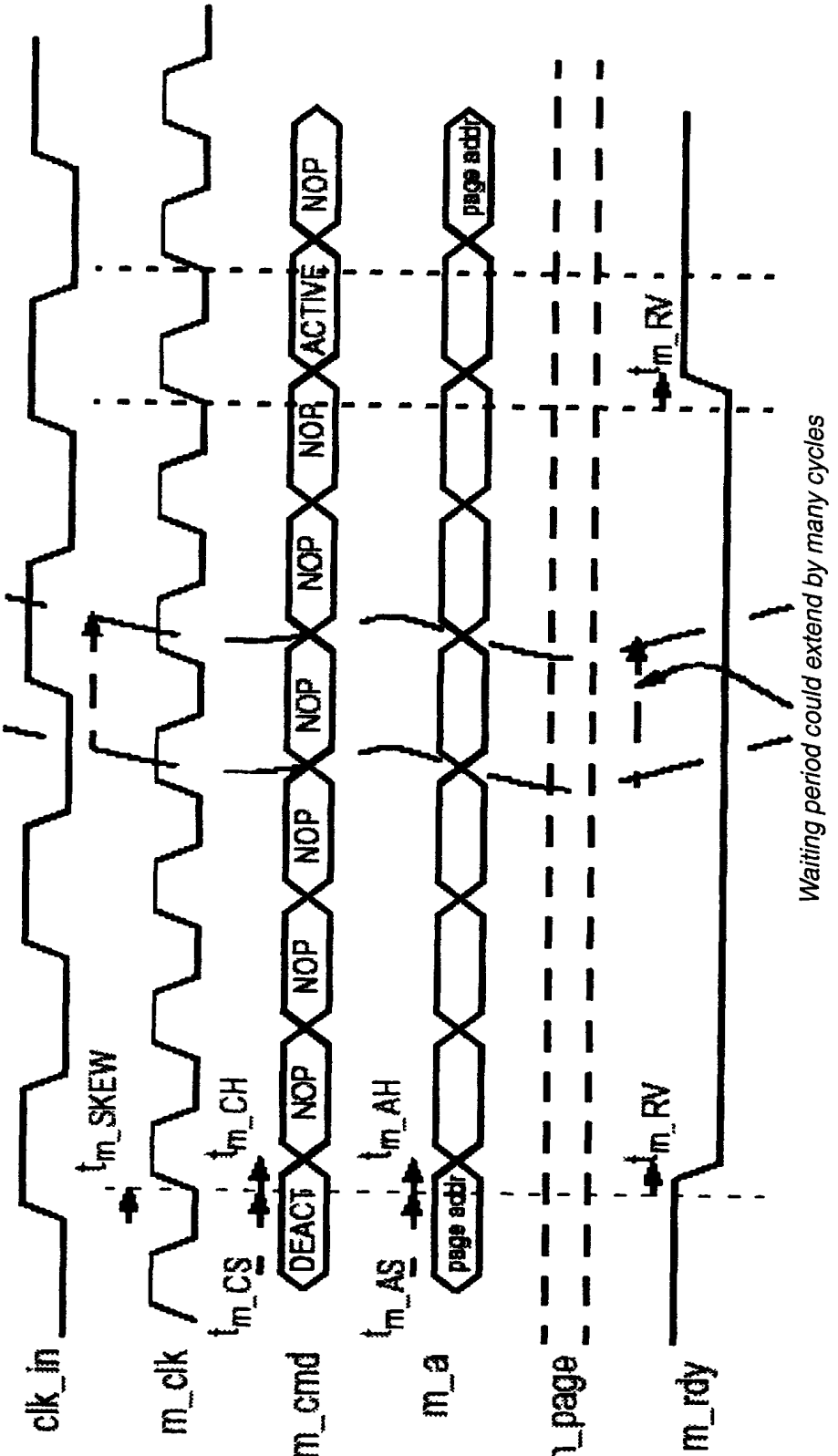

clk_in
m_clk
m_cmd
m_a
m_page
m_rdy
t_{m_SKEW}
t_{m_CS}
t_{m_CH}
t_{m_AS}
t_{m_AH}
t_{m_RV}
t_{m_RV}
DEACT
NOP
NOP
NOP
NOP
NOP
NOR
ACTIVE
NOP
page addr
page addr
Waiting period could extend by many cycles

FIG. 10

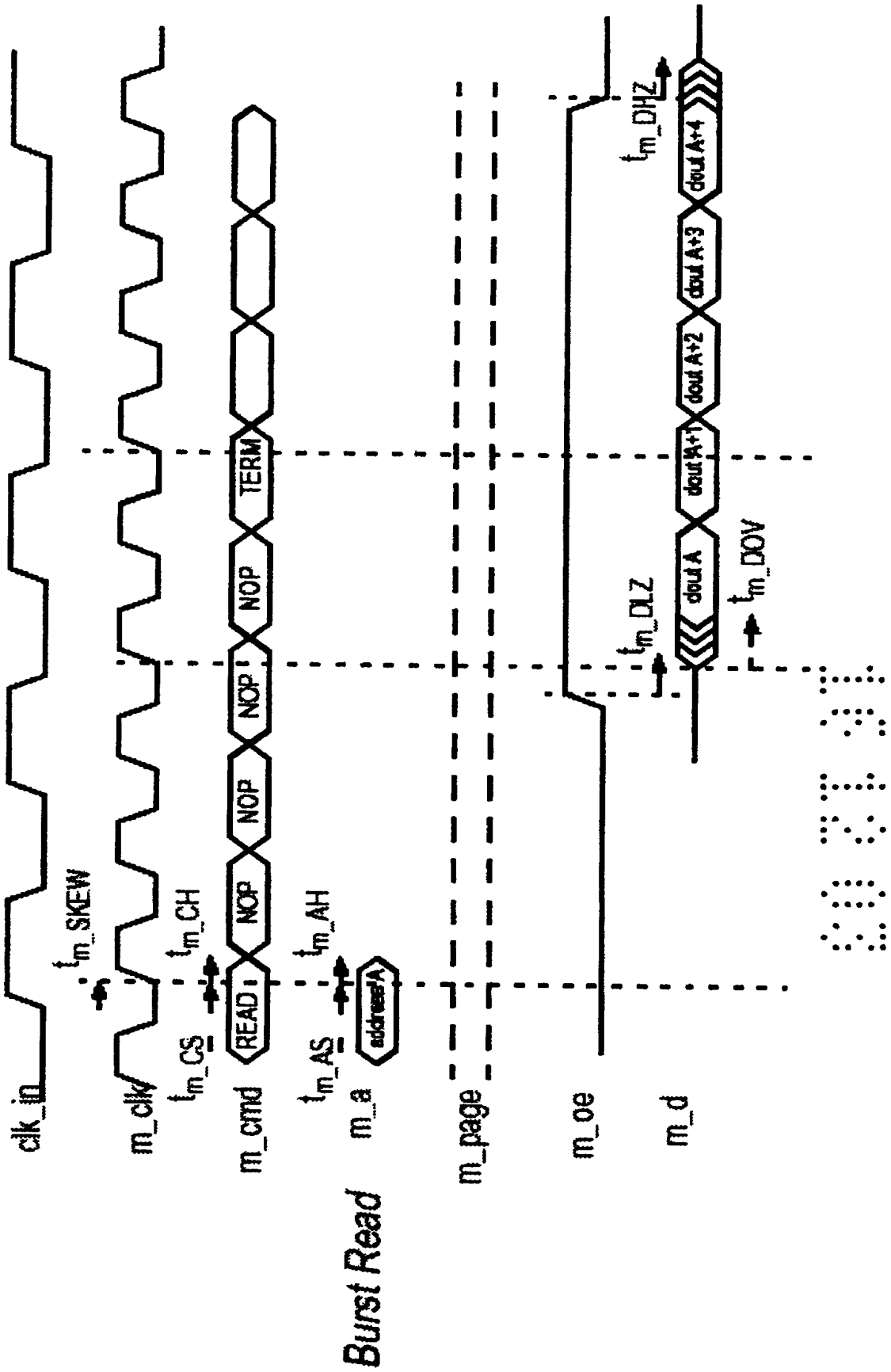

clk_in
m_clk
m_cmd
m_a
m_page
m_oe
m_d
Burst Read
t_{m_SKEW}
t_{m_CH}
t_{m_CS}
t_{m_AS}
t_{m_AH}
t_{m_DLZ}
t_{m_DOV}
t_{m_DHZ}
READ
NOP
NOP
NOP
NOP
TERM
address A
dout A
dout A+1
dout A+2
dout A+3
dout A+4

FIG. 11

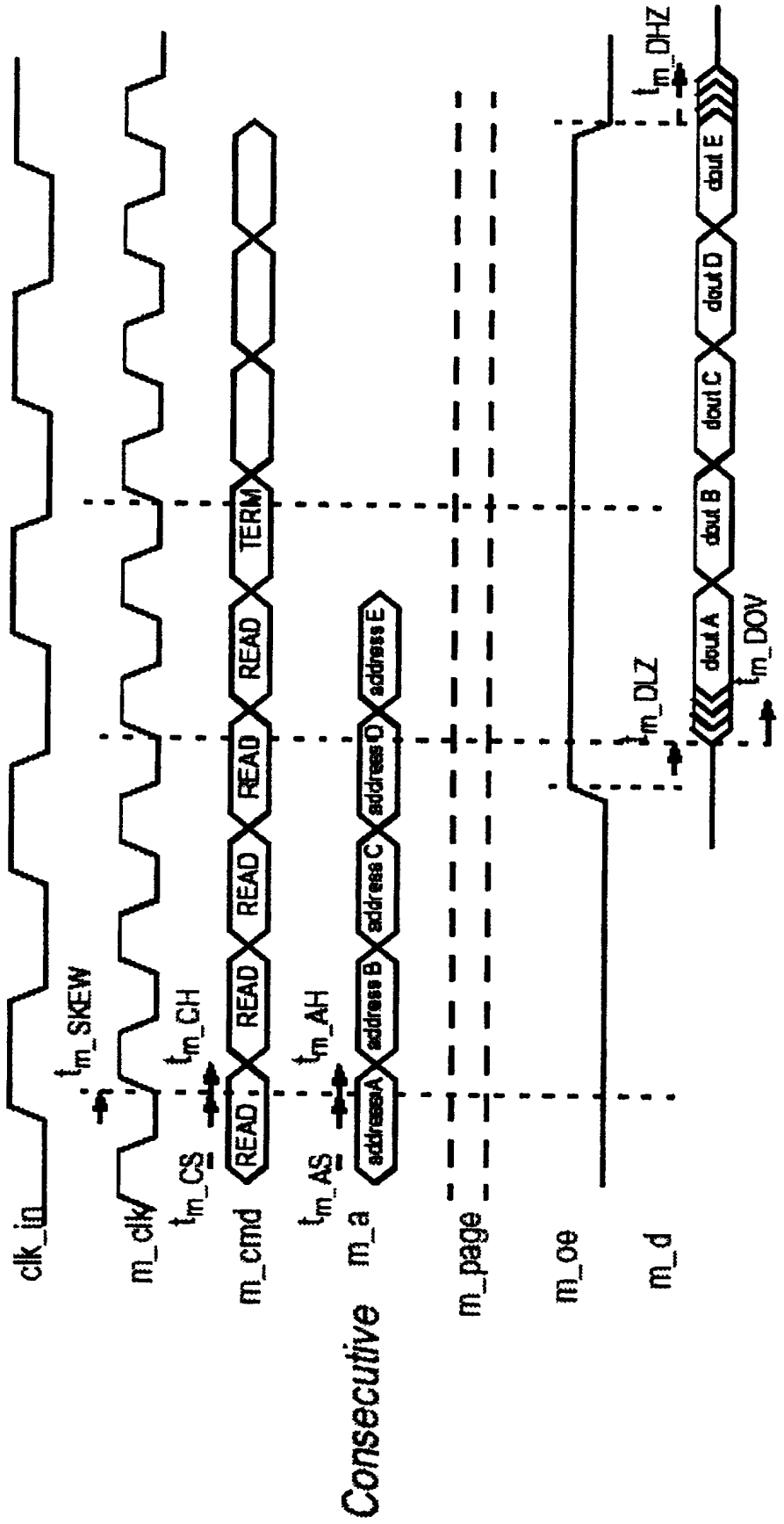

clk_in
m_clk
m_cmd
m_a
m_page
m_oe
m_d
Consecutive
t_{m_SKEW}
t_{m_CH}
t_{m_CS}
t_{m_AH}
t_{m_AS}
t_{m_DLZ}
t_{m_DOV}
t_{m_DHZ}
READ
TERM
addressA
address B
address C
address D
address E
dout A
dout B
dout C
dout D
dout E

FIG. 12

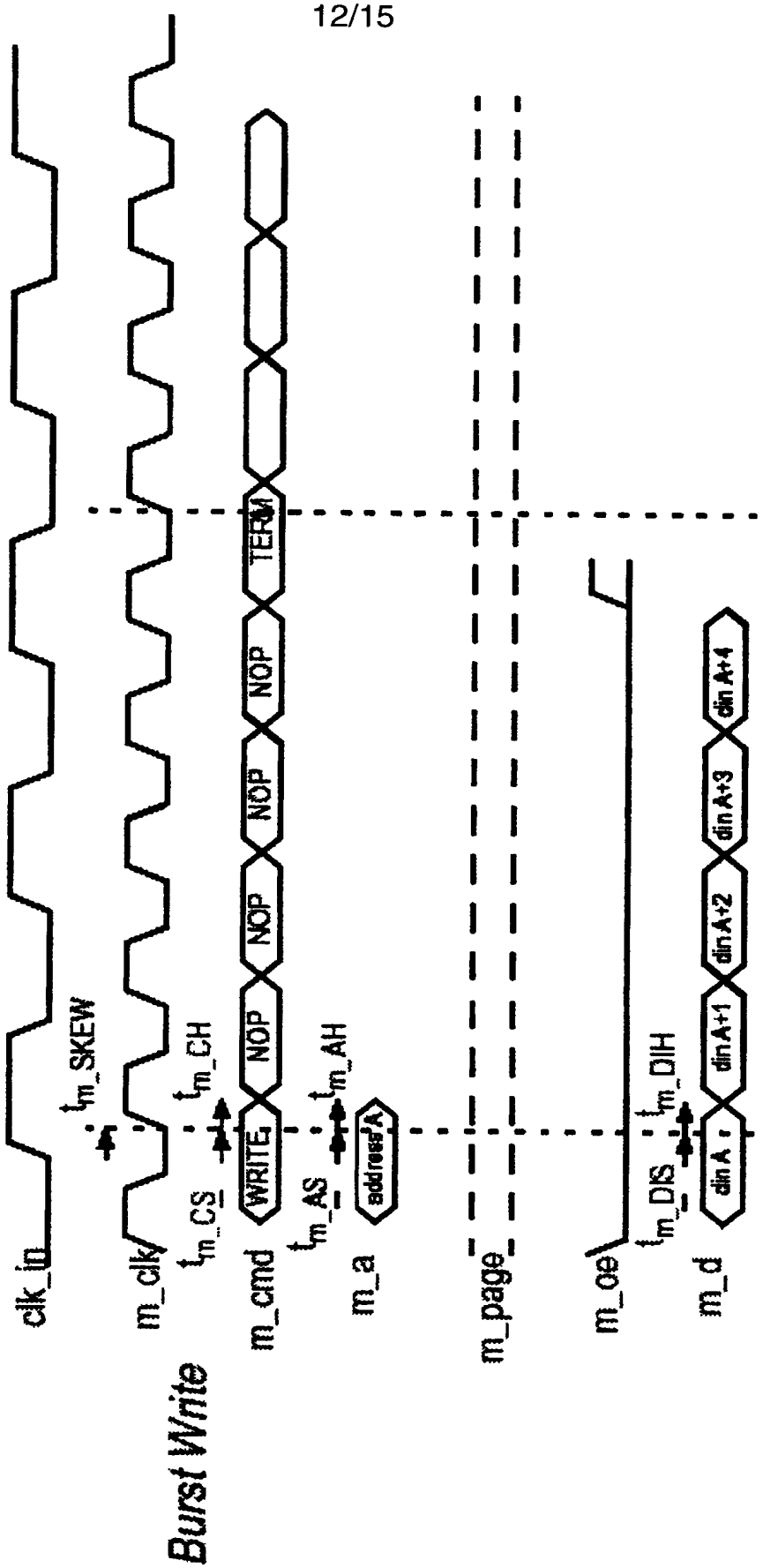

Burst Write
clk_in
m_clk
m_cmd
m_a
m_page
m_oe
m_d
t_{m_SKEW}
t_{m_CH}
t_{m_CS}
t_{m_AS}
t_{m_AH}
t_{m_DIS}
t_{m_DIH}
WRITE
NOP
NOP
NOP
NOP
TERM
address A
din A
din A+1
din A+2
din A+3
din A+4

FIG. 13

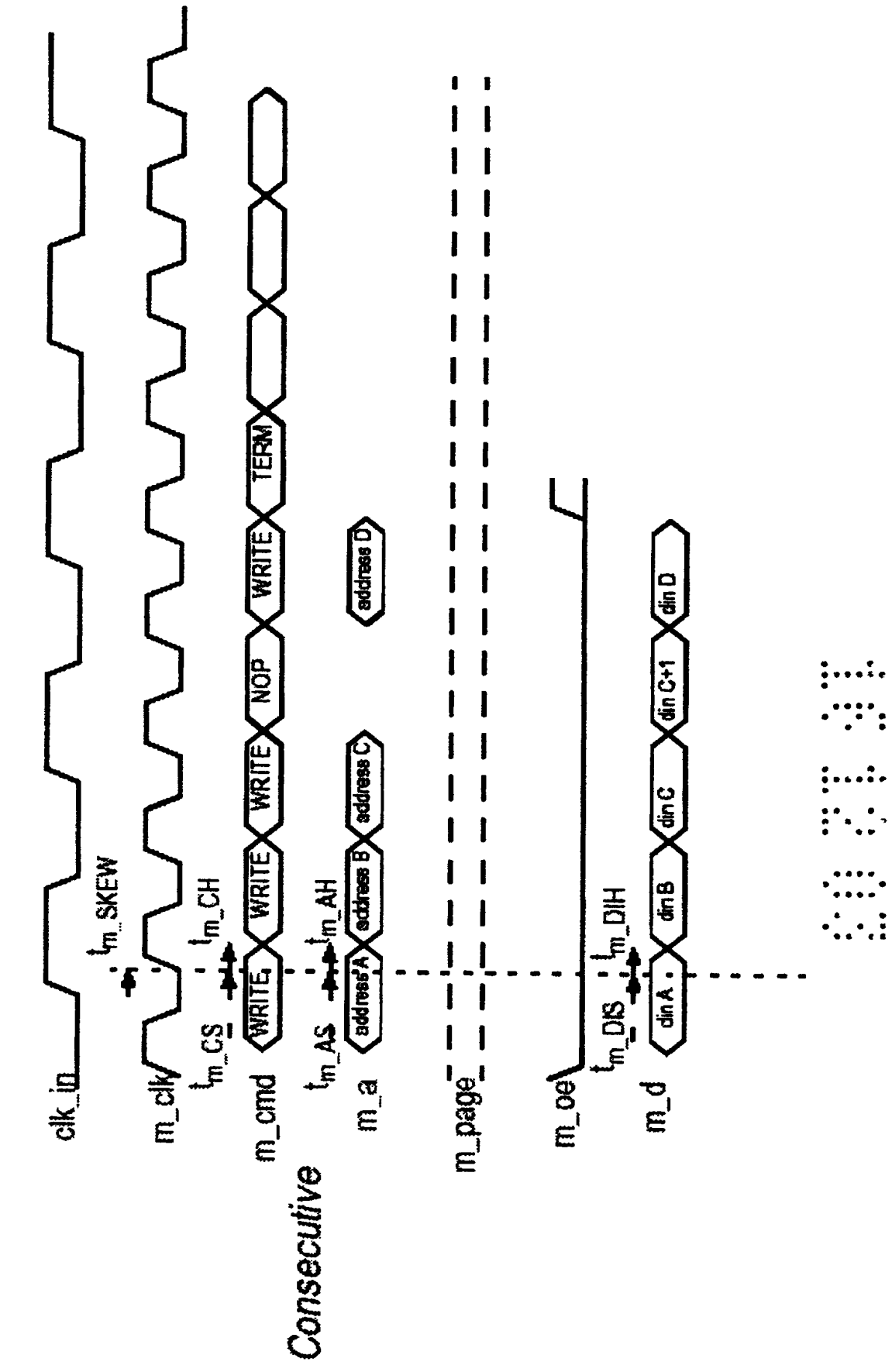

clk_in
m_clk
m_cmd
m_a
m_page
m_oe
m_d
t_{m_SKEW}
t_{m_CS}
t_{m_CH}
t_{m_AS}
t_{m_AH}
t_{m_DIS}
t_{m_DIH}
WRITE
WRITE
WRITE
NOP
WRITE
TERM
address A
address B
address C
address D
din A
din B
din C
din C+1
din D
Consecutive

FIG. 14

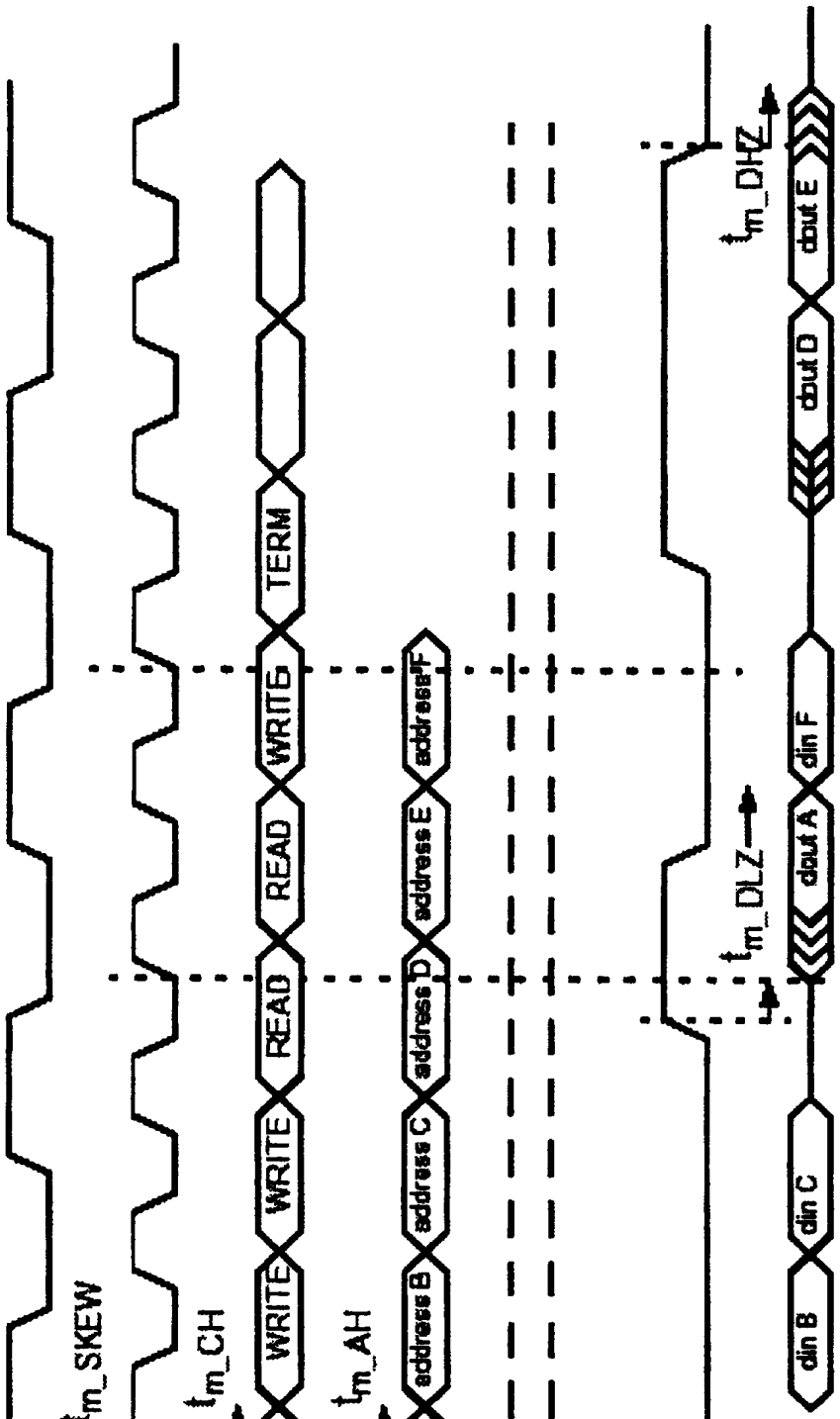
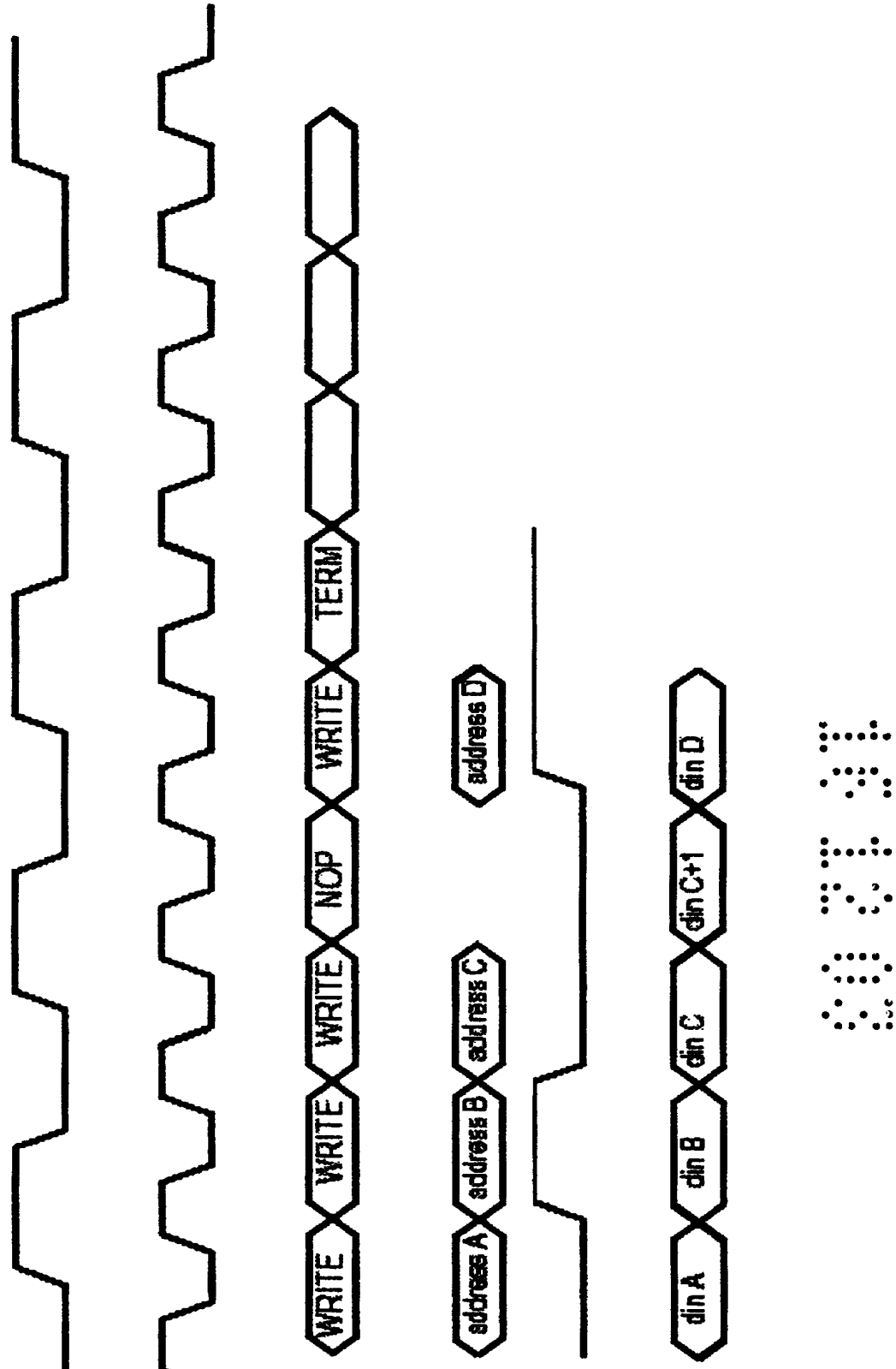

clk_in
m_clk
m_cmd
m_a
m_page
m_oe
m_d
t_{m_SKEW}
t_{m_CS}
t_{m_CH}
t_{m_AS}
t_{m_AH}
t_{m_DLZ}
t_{m_DOV}
t_{m_DHZ}
READ
WRITE
READ
WRITE
READ
WRITE
TERM
addressA
address B
address C
address D
address E
addressF
din B
din C
dout A
din F
dout D
dout E

FIG. 15


clk_in
m_clk
m_cmd
WRITE
WRITE
WRITE
NOP
WRITE
TERM
m_a
address A
address B
address C
address D
m_page
m_d
din A
din B
din C
din C+1
din D

HOST MEMORY INTERFACE FOR A PARALLEL PROCESSOR

FIELD OF THE INVENTION

The present invention relates to accessing data in a parallel processor including a memory array. Preferred embodiments of the present invention relate to accessing of data stored in memory connected to an array of processing elements in an active memory device by a host configured for connection with a conventional memory device.

BACKGROUND TO THE INVENTION

A simple computer generally includes a central processing unit CPU and a main memory. The CPU implements a sequence of operations encoded in a stored program. The program and data on which the CPU acts is typically stored in the main memory. The processing of the program and the allocation of main memory and other resources are controlled by an operating system. In operating systems where multiple applications may share and partition resources, the computer's processing performance can be improved through use of active memory.

Active memory is memory that processes data as well as storing it. It can be instructed to operate on its contents without transferring its contents to the CPU or to any other part of the system. This is typically achieved by distributing parallel processors throughout the memory. Each parallel processor is connected to the memory and operates on the memory independently of the other processing elements. Most of the data processing is performed within the active memory and the work of the CPU is thus reduced to the operating system tasks of scheduling processes and allocating system resources.

A block of active memory typically consists of the following: a block of memory, e.g. dynamic random access memory DRAM, an interconnection block and a memory processor processing element array. The interconnection block provides a path that allows data to flow between the block of memory and the processing element array. The processing element array typically includes multiple identical processing elements controlled by a sequencer. Processing elements are generally small in area, have a low degree of hardware complexity, and are quick to implement, which leads to increased optimisation. Processing elements are usually designed to balance performance and cost. A

simple more general-purpose processing element will result in a higher level of performance than a more complex processing element because it can be easily coupled to many identical processing elements. Further, because of its simplicity, the processing element will clock at a faster rate.

In any computer system, it is important that data can be made available to the processor as quickly as possible. In an active memory device, the complexity of the device means that data has to be accessed from the memory via the processing elements. Thus, the speed of access to the memory by a host processor is reduced. In addition, the added complexity that an active memory device bestows on a computer system means that additional complexity is added to the method of accessing data from the active memory device, which itself imparts additional complexity on the host processor.

In current systems, due to this additional complexity, a host connected to an active memory device has to be custom designed specifically for the active memory device. Thus, hosts configured for connection with one type of active memory device cannot be used with a different type of active memory device. Furthermore, hosts which have been designed for connection with conventional memory devices, such as standard SDRAM memories, cannot be connected to active memory devices at all. As such, considerable expense is incurred in the development of computer systems using active memory devices, since not only does the active memory device have to be designed and built, but also a complete host system to operate with it. Conventional memory devices are defined as any type of non-active memory devices which can be addressed by conventional memory command signals conforming to common industry standards.

Accordingly, it is an object of the present invention to provide a standard memory interface for an active memory device which permits different types of host processors to access the memory in the device.

It is a further object of the present invention to provide a memory interface for an active memory device for use with conventional host processors which are configured to connect to standard "non-active" memory devices, such as a standard SDRAM memory module.

SUMMARY OF THE INVENTION

In view of the foregoing and in accordance with one aspect of the present invention, there is provided a memory interface for a parallel processor having an array of processing elements, the memory interface being adapted to operate as follows:

to receive memory control signals and memory addresses from a host;

to apply at least a portion of the memory addresses to a memory connected to the processing elements; and

to apply control signals to the processing elements, such that in response the processing elements transfer data:

to and from the memory at the memory address; or

to and from the host; or

both; and

wherein the memory interface is adapted to connect to a host configured to access data in a conventional memory device, such that the host can access data in the memory.

The memory control signals and memory addresses may include a row address signal RAS, a row address, a column address signal CAS, a column address and a write enable signal WE.

The present invention further provides a memory interface for a parallel processor having an array of processing elements, the memory interface being adapted to operate as follows:

to receive memory control signals and memory addresses from a host;

to apply at least a portion of the memory addresses to a memory connected to the processing elements; and

to apply control signals to the processing elements, such that in response the processing elements transfer data:

to and from the memory at the memory address; or

to and from the host; or

both; and

wherein the memory control signals and memory addresses include a row address signal RAS, a row address, a column address signal CAS, a column address and a write enable signal WE.

Preferably, on receipt of a row address and a first configuration of memory control signals including a RAS assertion, the interface activates a page of data by transferring data from the row in the memory corresponding to the row address, into the processing elements.

Preferably, on receipt of a second configuration of memory command signals, the interface deactivates the page of data by transferring data from the processing elements into the row in the memory corresponding to the row address.

Preferably, on receipt of a column address and a third configuration of memory command signals including a CAS assertion and a WE deassertion, the interface transfers data from the activated page of data in the processing elements to the host, beginning with data from the column in the memory corresponding to the column address.

Preferably, on receipt of a column address and a fourth configuration of memory command signals including a CAS assertion and a WE assertion, the interface transfers data from the host to the activated page of data in the processing elements, beginning with data for the column in the memory corresponding to the column address.

The present invention further provides an active memory comprising a memory, an array of processing elements connected to the memory and a memory interface and methods of reading and writing to such a memory.

BRIEF DESCRIPTION OF THE DRAWINGS

A specific embodiment will now be described by way of example only and with reference to the accompanying drawings, in which:

Fig. 1 shows one embodiment of an active memory block in accordance with the present invention;

Fig. 2 shows one embodiment of the components of the active memory block in accordance with the present invention;

Fig. 3 shows one embodiment of control logic in the memory interface;

Fig. 4 shows one embodiment of a processing element in the active memory block in accordance with the present invention;

Figs. 5a and 5b show representations of the array of processing elements in accordance with the present invention;

Figs. 6a to 6c show different array address mappings in accordance with the present invention;

Figs. 6d to 6e show different mappings of bytes within a 32-bit word stored in host registers in the processing elements in accordance with the present invention;

Fig. 7 shows a state diagram for a finite state machine in the control logic in accordance with the present invention.

Figs. 8-15 are timing diagrams showing the operation of various memory commands.

DETAILED DESCRIPTION

Referring to Fig. 1, one embodiment of an active memory block in accordance with the invention is shown. Active memory block 100 includes a memory 106 and an PE array 110 of processing elements (PEs). Memory 106 is preferably random access memory (RAM), in particular DRAM. The PE array 110 communicates with memory 106 via an interconnection block 108. The interconnection block 108 can be any suitable communications path, such as a bi-directional high bandwidth path. A host 102, which in this case is a central processing unit CPU, communicates with the PE array 110 via memory interface 112. The memory interface 112 further communicates with the memory 106 via a DRAM control unit DCU 114. The memory interface includes conventional address, data and control lines.

Referring to Fig. 2, the active memory block 100 is shown connected to the host 102. The active memory block 100 comprises the memory 106, an array 110 of processing elements and the memory interface 112 having control logic 204 and a data register 206. The data register 206 is connected to the host 102 by a first data path 208 which is adapted to transfer high bandwidth data between the host 102 and the data register 206. The host 102 supplies a memory address 210 in the conventional way, using row (MSBs) and column (LSBs) addresses and RAS and CAS assertions, and other conventional memory access command signals 212 to the control logic 204. A READY signal 222 is generated by the control logic 204 and sent back to the host 102 to indicate that further command signals 212 can be sent.

The control logic 204 interprets the conventional memory access command signals 212 and the memory address 210 and generates an array address 214 from the column address of the memory address 210 and array control signals 216 which are sent to the PE array 110 and memory control signals 218 which are sent to the memory 106 via the DCU 114. The processing elements in the PE array 110 are configured to receive or send a row of data from or to the row in the memory 106 corresponding to the row address (MSBs) of the memory address 210. The PE array 110 is configured to respond to the array control signals 216 and the array address 214 to transfer data from the processing elements addressed by the array address 214. The data is transferred between the memory 106 and the PE array 110 via the interconnection block 108 and between the host 102 and the PE array 110 via the first and second data paths 208, 220 which are linked across the data register 206.

The control logic 204 also receives a page command signal 224 from the host 102 to determine which of two pages of data in the PE array 110 to address. The selection of the page is made via the array control signals 216.

Referring to Fig. 3, the control logic 204 is shown including an address register 302 for receiving the memory address 210 from the host 102, a mode register 304 for generating mode signals 312. A finite state machine FSM 306 receives the command signals 212 from the host 102 and the mode signals 312 from the mode register 304 and generates the memory control signals 218 and array control signals 216. Address transform logic 308 generates an array address 214 from the column address (LSBs) of the memory address 210 and sends it to the PE array 110, to address the appropriate processing elements in the PE array 110 corresponding to the array address 214 and the mapping of the addresses to the processing elements, as specified by the mode signals 312.

The contents of a mode register 304 is used to determine the data ordering in the PE array 110 and the memory 106 and sends mode signals 312 to the address transform logic 308 and the DCU 114 so that the address transform logic 308 can interpret and address the data in the PE array 110 correctly and the DCU 114 can address the data in the memory 106.

Referring to Fig. 4, a processing element 400 in the PE array 110 is shown comprising a DRAM interface 401 for connecting the memory 106 and the memory interface 112 with the processing element 400. Also included in the processing element 400 is a register file 406 between the result pipe 408 and processing logic 410. Data from the memory 106 is sent via the DRAM interface 401 to be processed in the processing logic 410 and moved between other processing elements in the PE array 110 via the result pipe 408. The DRAM interface 401 comprises host registers (H-registers) 402 and DRAM registers 404. The H-registers 402 receive from and send data to the memory interface 112 via the second data path 220.

The H-registers 402 are arranged in a first bank 451 and a second bank 452, each bank corresponding respectively to a first and second page of data to be stored in the H-registers 402 of all of the processing elements. The page to be addressed is determined by the page command signal 224 which is interpreted by the FSM 306 and sent to the PE array 110 with the array control signals 216. Thus, at any given time, two pages of data can be active in the PE array 110.

Every command issued to the interface, by a host processor or external I/O device is accompanied by a page select. The interface maintains a complete set of operational parameters for each page (for example the DRAM address used by the ACTIVE command). A page consists of four planes of DRAM bytes in the H-registers in each PE, or 1024 bytes. The data in the first plane is taken from the DRAM data at the page or row address supplied with the ACTIVE command described below. Once a page is held in the H-registers 402, burst reads and writes can take place as described below. The interface data input and output ports are 32 bits wide, and so the unit of data transfer during bursts is the 32 bit word. Each page contains 256 32 bit words, which are addressed with eight address bits. The mapping mode, described below, determines the way that each eight bit address maps to the bytes within the H registers.

The DRAM registers 404 receive data from and send data to the memory 102 at the row corresponding to the row address (MSBs) of the memory address 120 via the interconnection block 108. The data is received from the DRAM registers 404 and transferred between the memory interface 112 via one of the banks of H-registers 402, the

bank being specified by the array command signals 212. Each H-register can store one byte (8 bits) of data. Thus, a given processing element 400 can store a 32 bit word for each of the two pages.

Referring to Figs. 5a, 5b and 6a to 6c, a representation of the PE array 110 is shown having individual processing elements 400. In Fig. 5b, the first page 500 of data is shown with the H-registers 402 in the first bank 451 represented by four layers 501, 502, 503, 504 of H-registers 402. The second page of data is not shown, but in a similar way to the first page 500 uses four H-registers 402 in the second bank 452 and operates in a similar manner to the first page 500 as discussed below.

For the first page 500, each layer 501, 502, 503, 504 of H-registers corresponds to first, second, third or fourth H-registers in each processing element 400. For the PE array 110 shown in Fig. 5b, which has 16 rows and 16 columns, there are 256 processing elements and 1024 bytes of data in the first page 500.

Figs. 6a to 6c show different mappings of data in the PE array 110, the type of mapping being set or interpreted by the mode signals 312. The second data path 220 is 32 bits wide, so the corresponding unit of data transfer from the H-registers 402 to the data register 206 is a 32 bit word. There are 256 processing elements in the PE array 110 and therefore 256 32 bit words which are addressed by an array address 214 which is 8 bits wide.

In Fig. 6a, 32 bits of data are contained in each processing element 601, with 8 bits of data held in each of the four H-registers 402 in each processing element. This is referred to as 'word' mapping and is used for 32 bit processing element operations. Each array address corresponds to an entire processing element.

In Fig. 6b, 2×16 bits of data are contained in each processing element 601, 602, with 32 bits of data in total held across two H-registers 402 in each of two processing elements 601, 602. This is referred to as 'half-word' mapping and is used for 16 bit processing element operations. Thus, for each processing element, there are two mapped array addresses, with each array address corresponding to two different H-registers.

In Fig. 6c, 4×8 bits of data are contained in each processing element 601, 602, 603, 604, with 32 bits of data held across a single H-register 402 in each of four processing elements 601, 602, 603, 604. This is referred to as 'byte' mapping and is used for 8 bit processing element operations. Thus, for each processing element, there are four mapped array addresses, with each array address corresponding to a different H-register.

In addition to the aforementioned mappings of data in the PE array 110, the endianism of the data can be set by the host 102, i.e. the ordering of the bytes in each 32 bit word stored in the H-registers 402. There are two different orderings of bytes: big endian and little endian. Routines in the processing elements expect multi-byte words to be stored in the register file in a particular way and by convention big endian is the normal mode which means that the most significant byte of a multi-byte number is held in the lowest addressed register.

Big endian mode 670 is shown in Fig. 6d, which shows a lowest addressed register 671 containing a most significant byte 672 of a 32-bit word and a highest addressed register 673 containing a least significant byte 674. Little endian mode 680 is shown in Fig. 6e, which shows the lowest addressed register 671 containing the least significant byte 672 of a 32-bit word and the highest addressed register 673 containing the most significant byte 674.

The mapping and endian modes are specified by the host issuing a LOAD command (see below) and placing mode register fields (see Table 1 below) onto the memory address lines. The mode register fields are stored in the mode register 304 which sends the mode signals 312 to the address transform logic 308 so that the address transform logic can interpret the data in the PE array 110 appropriately.

Table 1: Mode register fields

Bits	Field	Comments
0 to 1	Mapping	0: word mapping 1: half-word mapping 2,3: byte mapping
2	Endianism	0: big-endian byte mapping 1: little-endian byte mapping

Referring to Fig. 7, a state diagram for the finite state machine FSM 306 is shown. As mentioned above, the FSM 306 receives conventional memory access command signals 212 from the host 101. The conventional memory access commands, which are interpreted by and implemented in the FSM 306 and shown in Fig. 7, are listed in Table 2 below.

Table 2: Command Functions and Encoding

Command value	RAS	CAS	WE	State
7	1	1	1	NOP 760
6	1	1	0	Burst Terminate 764
5	1	0	1	Read 756
4	1	0	0	Write 758
3	0	1	1	Active 754
2	0	1	1	Deactivate 752

In Table 2, the command signals 212 sent by the host 101 are the conventional memory access signals: RAS (Row Address Signal); CAS (Column Access Signal); and WE (Write Enable), which are interpreted by the FSM 306 as the states listed in Table 2 and shown in Fig. 7.

As can be seen from Fig. 7, the FSM 306 will remain in an idle state 702 and an active state 704 indefinitely until a command is issued by the host 101.

From the idle state 702, before data can be accessed, a page must be activated using the ACTIVE command 754 (see Table 1) to enter the active state 704 in which a page of 256

32-bit values has been activated in the H-registers 402 for reading and writing by the host 102. Activation consists of loading data from the memory 106 into the H-registers 402 of the processing elements according to the mapping scheme currently in force. The ACTIVE command 754 can take a variable amount of time, so a READY signal 222 signals to the host 102 that the ACTIVE command 754 has completed and the active state 704 has been entered. After an ACTIVE command 754 has been issued by the host 102, the command inputs will be ignored until after the READY signal 222 goes high indicating completion of the ACTIVE command 754. Once a page has been activated it remains active until a DEACTIVATE or PRECHARGE command is registered for that page.

Fig. 8 is a timing diagram illustrating the operation of the ACTIVE command. In figures 8-15, The various signals shown have the following significance.

Table 3: Signal Descriptions

Signal	In/Out	Description
m_clk	Out	Memory Port Timing Reference Clock. m_clk runs at twice the frequency of the master clock clk_in. Memory port transactions are timed relative to the rising edge of m_clk.
m_d[32]	In/Out	Memory interface data.
m_a[12]	In	Memory interface address.
m_cmd[3]	In	Memory interface command.
m_page	In	Memory interface page select: selects which page of H registers is activated by the current command.
m_ce	In	Memory interface enable: transaction only takes place when m_ce is active.
m_oe	In	Memory interface output enable: when (1), chip drives m_d out. When (0) m_d is high impedance.
m_rdy	Out	Memory interface ready: indicates completion of ACTIVE or DEACTIVATE command. A command should only be issued when m_rdy is high. After an ACTIVE or DEACTIVATE command is registered, no other commands are registered until the first clock edge after m_rdy goes high signalling completion.

In addition, the timing parameters used in figures 8-15 have the following significance.

Table 4: Timing Parameters

Timing	Description	Min (ns)	Max (ns)
t_{m_CS}	Command setup to clock	2.0	
t_{m_CH}	Command hold after clock	0.0	
t_{m_AS}	Address setup to clock	2.0	
t_{m_AH}	Address hold after clock	0.0	
t_{m_DIS}	Data in setup to clock	2.0	
t_{m_DIH}	Data in hold after clock	0.0	
t_{m_DOV}	Data output, clock to data valid	3.0	6.0
t_{m_DHZ}	Data output, m_oe to high Z		3.0
t_{m_DLZ}	Data output, m_oe to low Z	1.0	4.5
t_{m_RV}	m_rdy, clock to valid	3.0	6.0
t_{m_SKEW}	m_clk skew vs. clk_in	0	
t_{m_CLK}	Clock period	15	

From the active state 704, upon receipt of the READ command 756 (see Table 1), the FSM 306 enters a read state 706 in which data is transferred in a burst from the H-registers 402 along the second data path 220 to the data register 206 and from there to the host 102 along the first data path 120. Read accesses to the DRAM are burst-orientated, up to a maximum burst length of 256 32 bit words (a whole page). The first READ or WRITE command, described below, can be registered on the clock edge following the READY signal going high. The array address for beginning the read burst is taken from bits 7 to 0 (LSBs) of the memory address 210, corresponding to the column address received with the CAS assertion. If a read burst runs off the end of the page, then it wraps around back to the start of the page and continues automatically. Bursts may be any length, but if a burst continues for longer than a page of H-registers, namely 256 transfers, the data will be repeated.

Fig. 10 is a timing diagram illustrating the operation of a single burst READ command and Fig. 11 is a timing diagram illustrating the operation of the consecutive READ commands, illustrating the termination of prior READ bursts by subsequent READ commands.

From the active state 704, upon receipt of the WRITE command 758 (see Table 1), the FSM 306 enters a write state 704 in which data is transferred in a burst from the host 102 to the data register 206 along the first data path 120 and from the data register 206 to the H-registers 402 along the second data path 220. Write accesses to the DRAM are burst-orientated, up to a maximum burst length of 256 32 bit words (a whole page). The array address 214 for beginning the write burst is taken from bits 7 to 0 (LSBs) of the memory address 210, corresponding to the column address received with the CAS assertion. If a write burst runs off the end of the page, then it wraps around back to the start of the page and continues automatically. Bursts may be any length, but if a burst continues for longer than a page of H-registers, namely 256 transfers, the written locations will be repeated and overwritten.

Fig. 12 is a timing diagram illustrating the operation of a single burst WRITE command and Fig. 13 is a timing diagram illustrating the operation of the consecutive WRITE commands, illustrating the termination of prior WRITE bursts by subsequent WRITE commands.

READ and WRITE commands may be interleaved as illustrated in the timing diagram of Fig. 14. NOP commands may be inserted between consecutive READ commands or WRITE commands or interleaved READ and WRITE commands as illustrated in the timing diagram of Fig. 15, where a single NOP is inserted between the third and fourth WRITE commands to obtain a WRITE burst of 2 32-bit words. In Fig. 15, consecutive WRITE commands are shown addresses to alternate pages by toggling of the m_page signal. A burst to one page is terminated by any command to the other page.

A burst terminate command 764 (see Table 2) may be issued by the host 102 to terminate a data read or write burst and return the FSM 306 to the active state 704.

From the active, read or write states 702, 704 or 706, upon receipt of the DEACTIVATE or PRECHARGE command 752 (see Table 2), a page in the H-registers 402 is deactivated and its contents are returned to the memory 106 at the row corresponding to the row address part of the memory address 210 via the DRAM registers 404. The ACTIVE command can take a variable amount of time. Again, the READY signal is used to signal to the host that the DEACTIVATE or PRECHARGE command has completed. Thus, after a DEACTIVATE or PRECHARGE command 752 has been issued by the host 102, the command inputs will be ignored until after a READY signal 222 is asserted indicating completion of the DEACTIVATE or PRECHARGE command 752. If a page is activated by issuance of an ACTIVE command 754 and then no WRITE command 758 is issued, since no data has been written into the PE array 110 by the memory interface 112, the DEACTIVATE or PRECHARGE command 752 terminates immediately taking no action and asserting the READY signal 222.

Fig. 9 is a timing diagram illustrating the operation of the DEACTIVATE command.

The NOP command 760 see Table 2 is used to prevent unwanted commands from being registered during the idle, active, read or write states. Operations that are already in progress are not affected by issuance of the NOP command 760 by the host 102.

The LOAD command 762 (see Table 2) is a single-cycle command that can be issued at any time, except during activation and deactivation. Issuance of a LOAD command 762 by the host 102 will immediately terminate any read or write burst that is currently taking place. The LOAD command 762 causes the mode fields placed into the memory address lines by the host 101 to be loaded into the mode register 304.

It will of course be understood that the present invention has been described above purely by way of example and modifications of detail can be made within the scope of the invention.

CLAIMS

1. A memory interface for a parallel processor having an array of processing elements, the memory interface being adapted to operate as follows:

to receive memory control signals and memory addresses from a host;

to apply at least a portion of the memory addresses to a memory connected to the processing elements; and

to apply control signals to the processing elements, such that in response the processing elements transfer data:

to and from the memory at the memory address; or

to and from the host; or

both; and

wherein the memory interface is adapted to connect to a host configured to access data in a conventional memory device, such that the host can access data in the memory.

2. A memory interface circuit according to claim 1 further comprising a data register via which the processing elements transfer data to and from the host.

3. A memory interface according to claim 1 or claim 2 wherein the memory control signals and memory addresses include a row address signal RAS, a row address, a column address signal CAS, a column address and a write enable signal WE.

4. A memory interface for a parallel processor having an array of processing elements, the memory interface being adapted to operate as follows:

to receive memory control signals and memory addresses from a host;

to apply at least a portion of the memory addresses to a memory connected to the processing elements; and

to apply control signals to the processing elements, such that in response the processing elements transfer data:

to and from the memory at the memory address; or

to and from the host; or

both; and

wherein the memory control signals and memory addresses include a row address signal RAS, a row address, a column address signal CAS, a column address and a write enable signal WE.

5. A memory interface according to claim 3 or claim 4, adapted, on receipt of a row address and a first configuration of memory control signals including a RAS assertion, to activate a page of data by transferring data from the row in the memory corresponding to the row address, into the processing elements.

6. A memory interface according to claim 5, adapted, on receipt of a second configuration of memory command signals to deactivate the page of data by transferring data from the processing elements into the row in the memory corresponding to the row address.

7. A memory interface according to claim 5 or claim 6, adapted, on receipt of the first configuration of memory command signals, to apply the row address to the memory, to enable the page of data to be transferring from the memory into the processing elements.

8. A memory interface according to any one of claims 4-7, adapted, on receipt of a column address and a third configuration of memory command signals including a CAS assertion and a WE deassertion, to transfer data from the activated page of data in the processing elements to the host, beginning with data from the column in the memory corresponding to the column address.

9. A memory interface according to any one of claims 5-8, adapted, on receipt of a column address and a fourth configuration of memory command signals including a CAS assertion and a WE assertion, to transfer data from the host to the activated page of data in the processing elements, beginning with data for the column in the memory corresponding to the column address.

10. A memory interface according to claim 8 or claim 9 adapted, on receipt of the third or fourth configuration of memory control signals, to apply the column address to the processing elements, to identify the corresponding column in the memory in respect of

which the first data is to be transferred between the host and the activated page of data in the processing elements.

11. A memory interface according to any one of claims 5-10, adapted, on receipt of a further row address and the first configuration of memory control signals including a RAS assertion, to activate a second page of data by transferring data from the row in the memory corresponding to the further row address, into the processing elements.

12. An active memory comprising:

a memory;

an array of processing elements connected to the memory; and

a memory interface according to any preceding claim.

13. An active memory according to claim 12 further comprising one or more page registers in each processing element from or into which data is transferred to and from the memory and the host.

14. An active memory according to claim 13, wherein data from each memory address is transferred to and from the page registers in a single processing element.

15. An active memory according to claim 13, wherein data from different memory addresses is transferred to and from corresponding page registers in a plurality of processing elements.

16. A method of reading data from an active memory including a memory and an array of processing elements connected to the memory, comprising:

activating a page of data by transferring data from a row in the memory into the processing elements; and

reading data from the activated page of data in the processing elements and outputting it to a host.

17. A method of writing data to an active memory including a memory and an array of processing elements connected to the memory, comprising:

activating a page of data by transferring data from a row in the memory into the processing elements; and

inputting data from a host and writing it to the activated page of data in the processing elements.

18. A method according to claim 16 or claim 17 further comprising:

deactivating the page of data by transferring data from the processing elements into the row in the memory corresponding to the row address.

19

The Patent Office

INVESTOR IN PEOPLE

Application No:	GB 0228438.8	Examiner:	Ben Micklewright
Claims searched:	1-15	Date of search:	14 April 2004

Patents Act 1977 : Search Report under Section 17

Documents considered to be relevant:

Category	Relevant to claims	Identity of document and passage or figure of particular relevance	
X	1-3,5-15	US 5956274	(ELLIOTT) See e.g. column 5 line 44 to column 6 line 16 and figure 2
X	1-3,5-15	US 5953738	(RAO) See e.g. column 3 line 50 to column 4 line 50, columns 5-8, and figure 4
X	1-3,5-15	EP 0584783 A2	(TEXAS) See whole document, e.g. pages 4,5 and figure 1b
A	-	WO 01/01242 A1	(SUN) See e.g. the abstract, pages 13-22, and Figure 2
X	1-3,5-15	IEEE Transactions on VLSI Systems, Volume 4, Number 1, pages 56-69, 1996, "Programmable Active Memories: Reconfigurable Systems Come of Age", P Vuillemin et al; available from the Internet at http://citeseer.nj.nec.com/vuillemin96programmable.html	

Categories.

X	Document indicating lack of novelty or inventive step	A	Document indicating technological background and/or state of the art
Y	Document indicating lack of inventive step if combined with one or more other documents of same category	P	Document published on or after the declared priority date but before the filing date of this invention
&	Member of the same patent family	E	Patent document published on or after, but with priority date earlier than, the filing date of this application

Field of Search:

Search of GB, EP, WO & US patent documents classified in the following areas of the UKC[W]:

G4A

Worldwide search of patent documents classified in the following areas of the IPC[7]:

G06F, G11C

The following online and other databases have been used in the preparation of this search report:

WPI, EPODOC, PAJ, INSPEC, XPESP, Internet

PUB-NO: GB002396442A

DOCUMENT-IDENTIFIER: GB 2396442 A

TITLE: Host memory interface for a parallel processor

PUBN-DATE: June 23, 2004

INVENTOR-INFORMATION:

NAME	COUNTRY
KIRSCH, GRAHAM	GB

ASSIGNEE-INFORMATION:

NAME	COUNTRY
MICRON TECHNOLOGY INC	US

APPL-NO: GB00228438

APPL-DATE: December 5, 2002

PRIORITY-DATA: GB00221562A (September 17, 2002)

INT-CL (IPC): G06F015/78 , G11C007/10

ABSTRACT:

CHG DATE=20040629 STATUS=O>A memory interface 112 for a parallel processor 110 which has an array of processing elements and can receive a memory address and supply the memory address to a memory 106 connected to the processing elements. The processing elements transfer data to and from the memory at the memory address. The memory

interface can connect to a host 102 configured to access data in a conventional SDRAM memory device so that the host can access data in the memory. The invention also relates to an active memory which includes memory 106, an array of processing elements 110, and the memory interface 112. The memory interface enables a host which is configured to connect to conventional "non-active" memory devices to access an active memory, and also permits different types of host processors to access the memory in the active memory device.